Chapter 2 Solutions

Sidhanth Holalkere

January 29, 2021

1. In ϵ-greedy action selection, for the case of two actions and $\epsilon = 0.5$, what is the probability that the greedy action is selected.

   - At the start, there is a 0.5 chance that we immediately select the greedy action. If we don't select the greedy action, we randomly choose one of the two actions uniformly. The probability of randomly choosing the greedy action is $0.5 \times 0.5 = 0.25$. Therefore, the total probability that the greedy action is selected is $0.5 + 0.25 = 0.75$

2. *Bandit example* Consider a k-armed bandit problem with $k = 4$ actuibsm debited 1m 2m 3m abd 4, Consider applying to this problem a bandit algorithm using ϵ-greedy action selection, sample average action-value estimates, and initial estimates of $Q_1(a) = 0$, for all a. Suppose the initial sequence of actions and rewards is $A_1 = 1$, $R_1 = -1$, $A_2 = 2$, $R_2 = 1$, $A_3 = 2$, $R_3 = -2$, $A_4 = 2$, $R_4 = 2$, $A_5 = 3$, $R_5 = 0$. On some of thesesteps the ϵ case may have occurred, causing a action to be selected at random. On which time steps did this definitely occur? On which time steps could this possibly have occurred?

   - Starting at the beginning, A_1 could have been either since we don't know what the greedy action is yet. A_2 could have also been either since we still dont know what the greedy action is yet. A_3 definitely was an ϵ case since otherwise it would've chosen 2 which has the highest Q for now. A_4 could be either since it selected the greedy action (which could also have been randomly selected). And finally, A_5 is an ϵ step since it didn't choose the greedy action of 2.

3. In the comparison shown in Figure 2.2, which method will perform best in the long run in terms of cumultive reward and probability of selecting the best action? How much better will it be? Express your answer quantitatively.

   - Using $\epsilon = 0.01$ will perform better in the long run since once the algorithm's action-values approach their true values, it will pick the greedy (best) option around 9% more often than if you use $\epsilon = 0.1$.

4. If the step-size parameters, α_n, are not constant, then the estimate Q_n is a weighted average of previously received rewards with a weighting different from that given by (2.6). What is the weighting on each prior reward for the general case, analogous to (2.6), in terms of the sequence of step-size parameters?

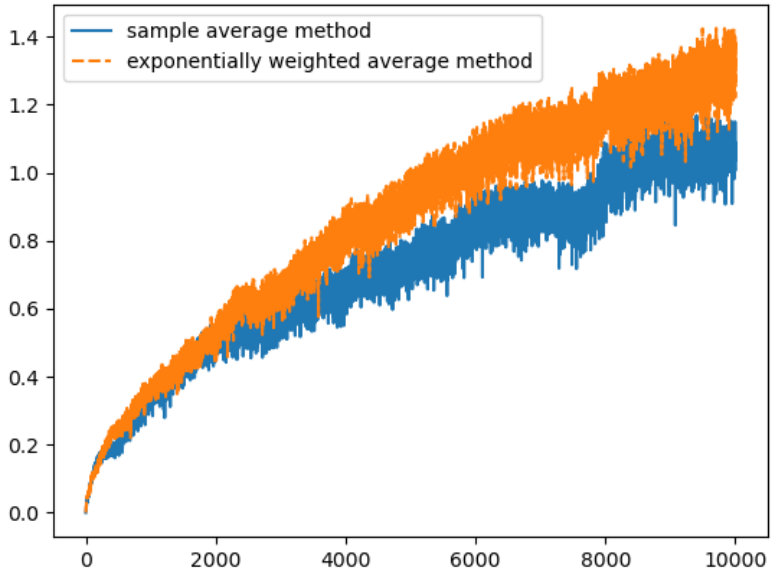
•

$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha_n[R_n - Q_n] && (1)\\
&= \alpha_n R_n + (1-\alpha_n)Q_n && (2)\\
&= \alpha_n R_n + (1-\alpha_n)[\alpha_{n-1}R_{n-1} + (1-\alpha_{n-1})Q_{n-1}] && (3)\\
&= \alpha_n R_n + (1-\alpha_n)\alpha_{n-1}R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1})Q_{n-1} && (4)\\
&= \alpha_n R_n + (1-\alpha_n)\alpha_{n-1}R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1})\alpha_{n-2}R_{n-2} + && (5)\\
&\ldots + (1-\alpha_n)(1-\alpha_{n-1})\ldots(1-\alpha_2)\alpha_1 R_1 + && (6)\\
&(1-\alpha_n)(1-\alpha_{n-1})\ldots(1-\alpha_1)Q_1 && (7)\\
&= \left(\prod_{i=1}^{n}(1-\alpha_i)\right)Q_1 + \sum_{i=1}^{n}\left(\left(\prod_{j=i+1}^{n}(1-\alpha_j)\right)\alpha_i R_i\right) && (8)
\end{aligned}
$$

5. Design and conduct and experiment to demonstrate the difficulties that sample-average methods have for nonstationary problems. Use a modified version of the 10-armed testbed in which all the $q_*(a)$ start out equal and then take independent random walks (say by adding a normally distributed increment with mean zero and standard deviation 0.01 to all the $q_*(a)$ on each step). Prepare plots like Figure 2.2 for an action-value method using sample averages, incrementally computed, and another action-value method using a constant step-size parameter, $\alpha = 0.1$. Use $\epsilon = 0.1$ and longer runs, say of 10,000 steps.

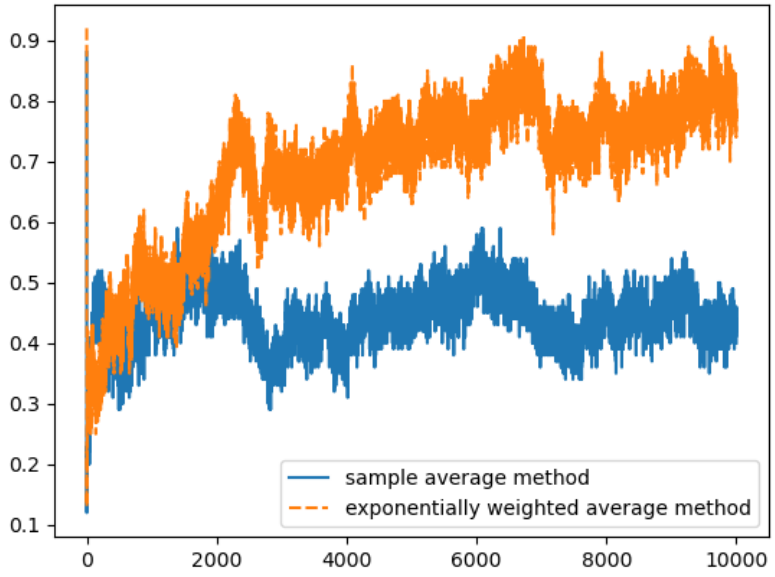
   - See 2-2.py and Figure 1 and Figure 2

Figure 1: The return of models using either sample-averages to calculate Q or a constant step-size (exponentially weighted average). The graph represents the average of 100 runs each.



6. *Mysterious Spikes* The results shown in Figures 2.3 should be quite reliable because they are averages over 2000 individual, randomly chosen 10-armed bandit tasks. Why, then, are there oscillations and spikes in the early part of the curve for the optimistic method? In other words, what might make this method perform particularly better worse, on average, on particular early steps?

Figure 2: The percentage of choosing the optimal action of models using either sample-averages to calculate Q or a constant step-size (exponentially weighted average). The graph represents the average of 100 runs each.



- Since the initial action-value predictions are optimistic, it takes longer for them to get closer to the true values, and therefore the algorithm spends more time thinking each action is "optimal" before identifying the truly optimal action. This makes it perform worse, on average, than using neutral action-values in earlier steps for the same reason.

7. *Unbiased Constant-Step-Size Trick* In most of this chapter we have used sample averages to estimate action values because sample averages do not produce the initial bias that constant step sizes do (see the analysis leading to (2.6)). However, sample averages are not a completely satisfactory solution because they may perform poorly on nonstationary problems. Is it possible to avoid the bias of constant step sizes while retaining their advantages on nonstationary problems? One way is to use a step size of $\beta_n \doteq \alpha/\bar{o}_n$ to process the nth reward for a particular action, where $\alpha > 0$ is a conventional constant step size, and $\bar{o}_n$ is a trace of one stat starts at 0: $\bar{o}_n \doteq \bar{o}_{n-1} + \alpha(1 - \bar{o}_{n-1})$, for $n \geq 0$. with $\bar{0}_0 \doteq 0$. Carry out an analysis like that in (2.6) to show that Q_n is an exponential recency-weighted average without initial bias.

- From (4.), we know that if the step size is not constant, the dependence, or bias, on the initial term Q_1 is as follows:

$$Q_{n+1} = \left(\prod_{i=1}^{n}(1-\beta_i)\right) Q_1 + ...$$

Which must equal 0 since for $i = 1$,

$$1 - \beta_1 = 1 - \frac{\alpha}{\overline{o}_1} \tag{9}$$

$$= 1 - \frac{\alpha}{\overline{o}_0 + \alpha(1 - \overline{o}_0)} \tag{10}$$

$$= 1 - \frac{\alpha}{0 + \alpha(1 - 0)} \tag{11}$$

$$= 1 - \frac{\alpha}{\alpha} \tag{12}$$

$$= 0 \tag{13}$$

Therefore there is no initial bias. As for it being an exponential recency-weighted average, we perform an analysis similar to (2.6)

$$Q_{n+1} = Q_n + \beta_n(R_n - Q_n) \tag{14}$$

$$= \beta_n R_n + (1 - \beta_n)Q_n \tag{15}$$

$$= \frac{\alpha}{\overline{o}_n} R_n + \left(1 - \frac{\alpha}{\overline{o}_n}\right) Q_n \tag{16}$$

$$Q_{n+1}\overline{o}_n = \alpha R_n + (\overline{o}_n - \alpha)Q_n \tag{17}$$

$$= \alpha R_n + (\overline{o}_{n-1} + \alpha(1 - \overline{o}_{n-1}) - \alpha)Q_n \tag{18}$$

$$= \alpha R_n + (\alpha + (1 - \alpha)\overline{o}_{n-1} - \alpha)Q_n \tag{19}$$

$$= \alpha R_n + (1 - \alpha)\overline{o}_{n-1} Q_n \tag{20}$$

$$= \alpha R_n + (1 - \alpha)(\alpha R_{n-1} + (1 + \alpha)\overline{o}_{n-2} Q_n) \tag{21}$$

$$= \alpha R_n + (1 - \alpha)\alpha R_{n-1} + (1 - \alpha)^2 \alpha R_{n-2} + \tag{22}$$

$$...(1 - \alpha)^{n-1} \alpha R_1 + (1 - \alpha)^n \overline{o}_0 Q_1 \tag{23}$$

$$= \sum_{i=1}^{n} \alpha(1 - \alpha)^{n-1} R_i \text{ Since } \overline{o}_0 = 0 \tag{24}$$

$$Q_{n+1} = \frac{1}{\overline{o}_n} \sum_{i=1}^{n} \alpha(1 - \alpha)^{n-1} R_i \tag{25}$$

Which is in the form of an exponential recency-weighted average.

8. *UCB Spikes* In Figure 2.4 the UCB algorithm shows a distinct spike in performance on the 11th step. Why is this? Note that for your answer to be fully satisfactory it must explain both why the reward increases on the 11th step and why it decreases on the subsequent steps. Hint: If $c = 1$, then the spike is less prominent.

   - Since it is a ten-armed bandit, the first ten steps are trying out each of the arms. Since 9/10 of the moves will not be optimal, the rewards (on average) will be low. Then, on the 11th step, since the $c\sqrt{\frac{\ln t}{N_t a}}$ is the same for all actions, it will pick what it believes to be the most optimal. But once the algorithm picks what it believes to be the optimal action again on the 11th step, it changes $N_t(a)$ (for what it believes is optimal) so the other actions have more uncertainty and will then be picked some more until the algorithm is confindent in the optimal choice.

9. Show that in the case of two actions, the soft-max distribution is the same as that given by the logistic, or sigmoid, function often used in statistics and artifial neural networks.

   - Say we have two actions a and b, with preferences $H_t(a)$ and $H_t(b)$. This means that the probabilities of selecting a and b, respectively, are $\frac{e^{H_t(a)}}{e^{H_t(a)} + e^{H_t(b)}}$ and $\frac{e^{H_t(b)}}{e^{H_t(a)} + e^{H_t(b)}}$. Taking the first one

($\pi(a)$) we can simplify as follows:

$$\pi(a) = \frac{e^{H_t(a)}}{e^{H_t(a)} + e^{H_t(b)}} \tag{26}$$

$$= \frac{e^0}{e^0 + e^{H_t(b)-H_t(a)}} \tag{27}$$

$$= \sigma(H_t(a) - H_t(b)) \tag{28}$$

$$\tag{29}$$

Similarly, $\pi(b) = \sigma(H_t(b) - H_t(a))$. These show that the softmax distribution is the same as the logistic one (of the difference in preference) in the case of two actions.

10. Suppose you face a 2-armed bandit task whose true action values change randomly from time step to time step. Specifically, suppose that, for any time step, the true values of actions 1 and 2 are respectively 0.1 and 0.2 with probability 0.5 (case A), and 0.9 and 0.8 with probability 0.5 (case B). If you are not able to tell which case you face at any step, what is the best explanation of success you can achieve and how should you behave to achieve it? Now suppose that on each step you are told whether you are facing case A or case B (although you still don't know the true action values). This is an associative search task. What is the best expectation of success you can achieve in this task, and how should you behave to achieve it?

    - If we are not able to tell which case we are in, the expected return would be 0.5 (the average of all actions over cases). This can be achieved by acting randomly. If we knew what case we were in we would always choose action 2 in case A and action 1 in case B which would have an average return of 0.55. One simple way to achieve this is to have effectively two bandits that we can switch in between depending on which case we know we are in.

11. *(programming)* Make a figure analogous to Figure 2.6 for the nonstationary case outlined in Exercise 2.5. Include the constant-step-size ϵ-greedy algorithm with $\alpha = 0.1$. Use runs of 200,000 steps and, as a performance measure for each algorithm and parameter settings, use the average reward over the last 100,000 steps.